IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 12, Issue 5, May 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.379

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 5, May 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1205332 |

Transforming Assessment: A Microservices Approach in Online Assessment Platforms

Atharva Mandpe, Arya Pathak, Omkar Lolage, Nishant Lanjewar, Aditya Lad

Assistant Professor, Department of Computer Engineering, Vishwakarma Institute of Technology, Pune, India

U.G. Student, Department of Computer Engineering, Vishwakarma Institute of Technology, Pune, Maharashtra, India

U.G. Student, Department of Computer Engineering, Vishwakarma Institute of Technology, Pune, Maharashtra, India

U.G. Student, Department of Computer Engineering, Vishwakarma Institute of Technology, Pune, Maharashtra, India

U.G. Student, Department of Computer Engineering, Vishwakarma Institute of Technology, Pune, Maharashtra, India

ABSTRACT: This paper explores the development and deployment of an advanced online coding assessment platform within the contemporary talent acquisition landscape. Emphasizing the necessity for innovation in evaluation processes spanning recruitment, skills enhancement, and performance appraisal, the platform's architecture is meticulously designed for scalability and security. The microservices-based infrastructure ensures heightened scalability, while robust security measures are integrated into the system. Leveraging Angular for the frontend and ASP.Net Core for the backend, the platform features three key microservices:

Central to the platform's design is the focus on secure code execution, ensuring isolation and protection against potential vulnerabilities. Security measures including containerization, network segmentation, and resource constraints are implemented to fortify the system against external threats. Through this paper, the intricate details of the system architecture and implementation are elucidated, showcasing its efficacy in addressing the evolving needs of online assessment in a secure and scalable manner.

KEYWORDS: System Design, Microservices, Kubernetes, Angular, ASP.Net Core, Coding assessment platform

I. INTRODUCTION

The quest for efficient and reliable talent evaluation methods has become paramount for organizations across various industries. Traditional assessment methods, often paper-based or involving in-person proctoring, can be time-consuming, resource-intensive, and prone to logistical challenges. Online assessment platforms have emerged as a compelling solution, offering several advantages:

Efficiency: They streamline the evaluation process by automating tasks such as question delivery, submission collection, and scoring (where applicable).

Scalability: They can accommodate many assessments and participants simultaneously, making them ideal for organizations with high evaluation demands.

Flexibility: Online assessments can be conducted remotely, offering greater flexibility for both organizations and participants in terms of time and location.

Data-driven Insights: Online platforms facilitate the collection of valuable data on participant performance, which can be used for informed decision-making and skill gap identification.

However, security and scalability remain key concerns when choosing an online assessment platform due to the inherent risk associated with untrusted code submissions, sensitive information such as participant data and assessment content needs robust protection against unauthorized access and data breaches. Additionally, the platform must seamlessly scale to accommodate a growing user base and diverse assessment needs. This paper presents a secure and scalable online code execution system architecture that leverages microservices and containerization for enhanced security and isolation.

II. RELATED WORK

Mistry et. al. [2] discussed the pressing need for secure and efficient online examination systems in response to the challenges posed by the COVID-19 pandemic's shift to remote learning. The work emphasizes the importance of implementing measures to prevent cheating while enabling rapid assessment and automatic grading. Addressing the limitations of traditional online assessment methods, the authors propose methodologies for a Python-based web

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 5, May 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1205332 |

application, incorporating features like full-screen mode and streamlined data management to enhance user experience and exam integrity. They advocate for continuous adaptation to the evolving distance education landscape, underscoring the importance of automated grading to reduce examiner fatigue and ensure consistency in assessment outcomes. While offering innovative solutions, further research is needed to evaluate the scalability and effectiveness of these methodologies in real-world educational contexts, ensuring the ongoing relevance and efficacy of online examination systems.

Dumal et. al. [3] presented a novel approach to address the challenges in evaluating subjective and text-based answers within e-learning systems through their work. While existing systems primarily focus on multiple-choice questions, the authors recognized the need for a comprehensive solution capable of assessing short and essay-type questions while providing immediate and reliable feedback to learners. Their research aims to optimize the evaluation process by automating the assessment of text-based answers, thereby enhancing efficiency and effectiveness in e-learning environments. By developing a system that allows tutors to create assignments, add questions to a question bank, and obtain final scores automatically, the authors provide a holistic solution for academic institutes to streamline their assignment evaluation processes. This study contributes to the advancement of adaptive and automated online assessment evaluation systems, facilitating improved learner engagement and academic outcomes in e-learning contexts. Huo et. al [4] introduced a lightweight and flexible SQL assessment platform tailored for teaching and learning purposes, as outlined in their work. The paper addressed the limitations of existing SQL test platforms, which often require complex environmental setups and lack cross-platform compatibility. Leveraging frontend database technology, the proposed platform utilized JavaScript engine and Node.js server to offer a user-friendly interface for SQL programming exercises. Students benefit from immediate query results feedback, while teachers can efficiently monitor student progress through backend server functionalities. Moreover, the platform incorporated browser fingerprinting techniques to detect plagiarism and cheating behaviours, ensuring academic integrity. This contribution fills a gap in the availability of lightweight and pedagogically focused SQL assessment platforms, offering a valuable resource for database course learning and teaching.

There are still some challenges, though. Traditional systems have trouble keeping up with lots of users, which can make them slow and crash. Security weaknesses, confusing interfaces, and unreliable data storage are other problems. To improve things, future systems should be designed to be secure and handle lots of users, be easy to use, and have strong data management. We also need to constantly look for and fix security vulnerabilities and monitor system performance to make sure online assessment platforms are effective and reliable.

III. TECHNOLOGY STACK

The proposed platform is meticulously constructed upon a robust technology stack, carefully chosen to ensure efficiency, performance, and security:
Frontend: Angular is chosen for its exceptional capabilities in single-page application (SPA) development. Its extensive community support and rich ecosystem of libraries and tools contribute to efficient development and maintenance. Angular's features like dependency injection, routing, and component-based architecture promote clean and maintainable code, facilitating a user-friendly and responsive interface for the platform.

Backend: ASP.Net Core provides a cutting-edge, platform-agnostic framework to efficiently build and manage the backend services. Its focus on performance and security aligns perfectly with the platform's objectives. ASP.Net Core leverages modern .NET features and offers robust functionalities for data access, API development, and authentication, making it an ideal choice for building the secure and scalable backend of the platform.

Containerization: Docker containers are leveraged to package and deploy individual microservices. This ensures consistent and isolated environments throughout the development, testing, and production stages. By containerizing each microservice, developers can guarantee that the service runs in a predictable environment with all necessary dependencies included. This simplifies deployment and management, as containers can be easily replicated and scaled across different environments.

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 5, May 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1205332 |

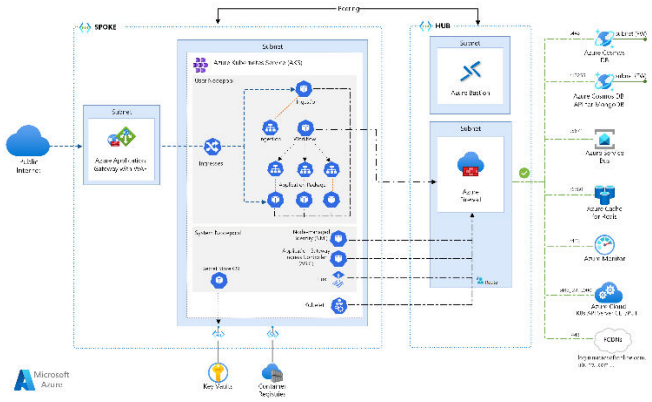


Figure 1. Kubernetes System Diagram

Orchestration: Kubernetes, a powerful container orchestration platform, is employed to manage the containerized microservices. It effectively automated deployment, scaling, and load balancing, facilitating a scalable and robust infrastructure. This allows for efficient management of the platform's microservices, ensuring they are deployed to the correct nodes, scaled based on resource demands, and load-balanced for optimal performance.

IV. SYSTEM ARCHITECTURE

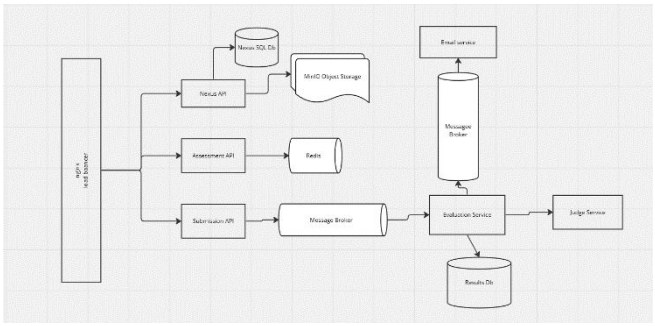


Figure 2. System Architecture

Fig. 2 shows the architecture of the proposed system. Here's a breakdown of the components and their connections:
1. Load Balancer: This is the entry point for incoming requests. It distributes the incoming traffic across multiple instances of Nexus API.
2. Nexus API: This API serves as an interface for various functionalities related to the assessment system. It interacts with a Nexus SQL Database and MinIO Object Storage.
3. Assessment API: Another API involved in the system, responsible for managing assessment-related tasks.
4. Submission API: Handles submissions made by users related to assessments.
5. Message Broker: RabbitMQ acts as a middleware for communication between different components of the system. It connects to Redis and facilitates communication with Evaluation Service.
6. Redis: In-memory data structure store used as a caching layer or for transient data storage.
7. Evaluation Service: This service evaluates submissions or assessments. It receives messages through the Message Broker and interacts with a results Database.
8. Judge Service: Likely responsible for making final judgments or decisions based on evaluations. It interacts with the Results Database.
9. Email Service: An external service that handles email notifications, possibly related to assessment outcomes or other system events with SMTP protocol.
10. Results Database: Stores the results or outcomes of evaluations. It is accessed by both the Evaluation Service and the Judge Service.

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 5, May 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1205332 |

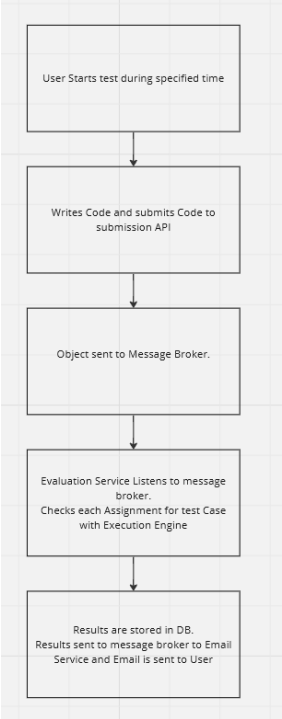


Figure 3. Examinee Workflow Flowchar

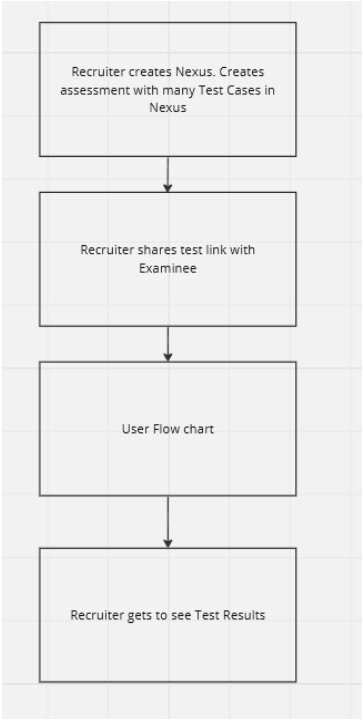


Figure 4. Recruiter Workflow Flowchart

The process outlined in Fig. 3 illustrates a comprehensive approach to conducting time-bound tests, specifically tailored for evaluating code submissions in programming or software development assessments. It begins with users initiating tests within predefined time frames, ensuring structured evaluation procedures. Participants then engage directly with the assessment platform, submitting their code solutions through a Submission API. These submissions are transmitted to a central Message Broker, which serves as the communication hub within the system architecture.
A critical component, the Evaluation Service, listens to the Message Broker for incoming submissions and employs an Execution Engine to assess the performance and correctness of each code submission against predefined test cases.
The results of these evaluations are stored in a dedicated database for future reference and analysis.

Furthermore, mechanisms for result dissemination ensure efficient communication of assessment outcomes. Evaluated results are routed through the Message Broker to an Email Service, which generates and sends email notifications to users regarding their test outcomes. This process enables timely feedback delivery, empowering participants to gauge their performance effectively and facilitating ongoing refinement of the assessment process based on insights derived from the stored data.

Fig. 4 presents a recruitment or evaluation framework operated through the Nexus platform. Recruiters utilize Nexus to craft assessments containing multiple test cases, which are then shared with examinees via unique links. Examinees submit their solutions through Nexus, and recruiters access the results for comprehensive review and analysis. This process ensures efficient recruitment procedures, promoting informed decision-making while maintaining a user-friendly experience for all participants.

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 5, May 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1205332 |

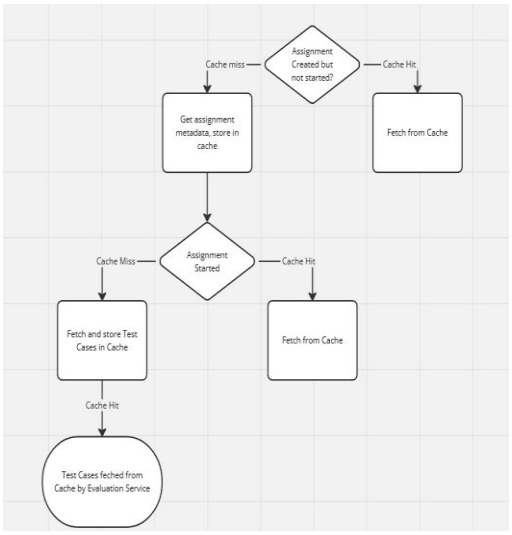


Figure 5. Caching Strategy

Fig. 5 illustrates a caching mechanism within a system architecture, highlighting its role in optimizing data retrieval and system performance. Initially, the system checks for cache misses, leading to the retrieval and storage of assignment metadata and test cases. This proactive approach minimizes data retrieval overhead, ensuring vital details are readily accessible for future use. Subsequently, during assignment commencement, the system assesses cache states and fetches test cases if needed, enhancing responsiveness and efficiency. In subsequent interactions, cache hits facilitate swift data retrieval, while cache misses prompt fetching from the original source, optimizing evaluation procedures. The Evaluation Service accesses pre-cached test cases directly, expediting assessments without latency from external data sources. This caching mechanism contributes to overall system performance and efficiency.

Data Persistence: Local persistent volumes attached to the Kubernetes cluster ensure persistent and reliable data storage and retrieval for the services. This guarantees that organization data, user submissions, and assessment details are preserved even if container instances are restarted or scaled. Persistent volumes ensure that critical data is not lost when containers are recreated, providing reliability and data durability for the platform.

Monitoring and Logging: System monitoring and logging facilitate security incident detection and response. The system actively monitors for suspicious activity and logs events for potential analysis, allowing for prompt identification and mitigation of security threats.

V. USER INTERFACE

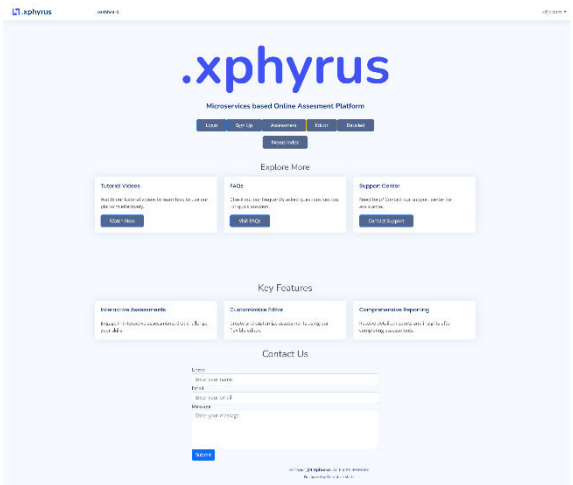


Figure 6. Home Page of the web platform

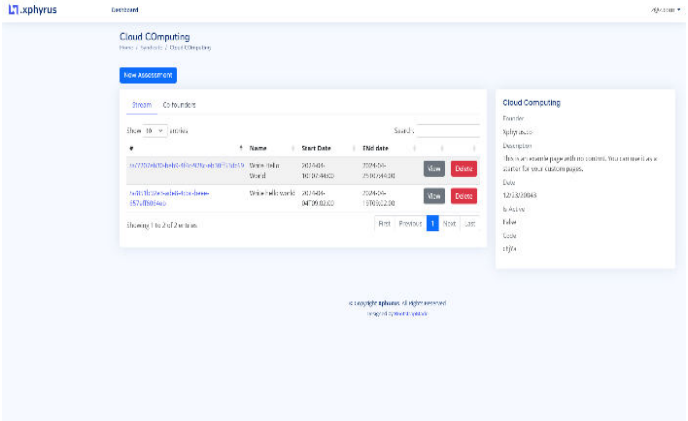


Figure 7. Nexus Index

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 5, May 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1205332 |

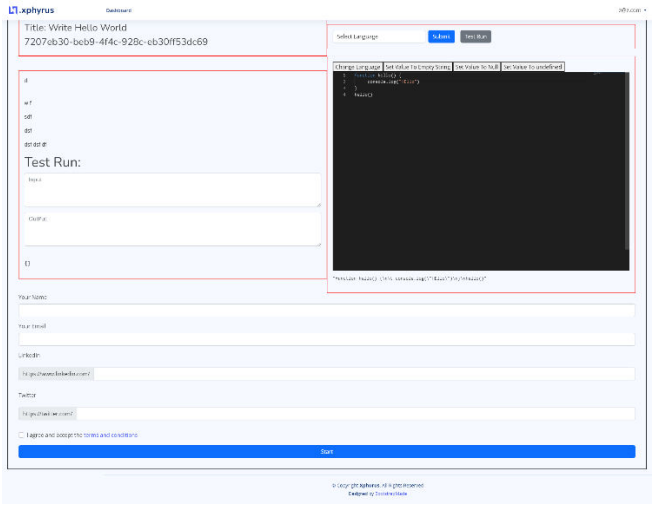
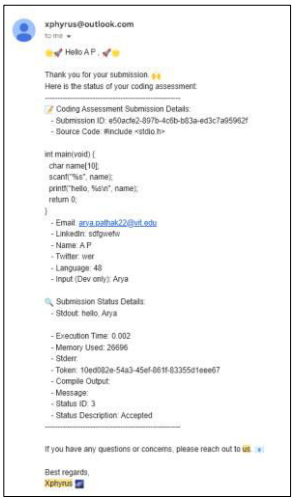


Figure 8. Online editor to write and submit the assessment Figure



9. Successful Submission Email with Result

VI. METHODOLOGY

The platform's deployment strategy caters to different stages, ensuring smooth development, testing, and production environments:

Initial Deployment: The platform is initially deployed within a local Kubernetes cluster comprising two nodes. This setup facilitates development, testing, and demonstration purposes. Developers can iterate on functionalities, conduct thorough testing, and showcase the platform's capabilities in a controlled environment. The local cluster allows for rapid development cycles and efficient debugging.

Production Deployment: For production deployments, a cloud-based Kubernetes cluster with autoscaling capabilities can be employed. This allows the platform to dynamically adjust resource allocation based on user traffic, ensuring optimal performance under varying loads and user bases. Azure Kubernetes Service (AKS) offers managed Kubernetes services, simplifying deployment and cluster management in the cloud. Autoscaling capabilities automatically scale the platform's microservices up or down based on resource demands, optimizing resource utilization and cost efficiency. Load balancer distributing traffic among worker nodes.

Analysis of Performance Results:
The performance testing revealed that the system exhibited good scalability as the number of concurrent users increased. The response time and round-trip time remained within acceptable limits for low to moderate user loads (up to 50 concurrent users). However, for higher user loads (above 50 concurrent users), the response time and round-trip time started to increase. This indicates that the system might require further optimization for handling extremely high loads.

Here's a breakdown of potential observations from the results (note that specific details would depend on the actual data):
Low to Moderate Loads: For a small number of concurrent users, the system resources (CPU, memory) were sufficient to handle the execution requests efficiently, resulting in fast response times and round-trip times.
High Loads: As the number of concurrent users increased, the demand for resources also grew. The system might have encountered resource constraints, leading to increased queuing times for job execution and, consequently, higher response times and round-trip times.

Optimization Strategies:
Based on the performance analysis, several strategies can be employed to optimize the system for handling high loads:
Horizontal Pod Autoscaling: Kubernetes offers horizontal pod autoscaling (HPA) functionality. This feature can be implemented to automatically scale the number of Submission Runner pods based on resource utilization. When resource usage increases beyond a predefined threshold, HPA can dynamically provision additional pods to handle the increased load, ensuring efficient resource allocation and improved performance.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 5, May 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1205332 |

Resource Optimization: Analysing the resource consumption of different algorithms can help identify opportunities for optimization. For instance, if certain algorithms exhibit high memory usage, language-specific container images with minimal dependencies can be created to reduce memory footprint and improve overall efficiency

VII. EXPERIMENTAL RESULTS

Table 1 presents a comparison of performance metrics for three configurations of a code execution engine: two monolithic setups and one Kubernetes cluster configuration. The monolithic setups vary in thread count and memory size but demonstrate similar minimum and maximum response times, indicating consistent performance under different loads. In contrast, the Kubernetes cluster, while showing slightly higher minimum response times, exhibits significantly higher maximum response times, suggesting potential performance challenges under certain conditions due to its distributed nature. However, the Kubernetes cluster displays better scalability and consistency, particularly evident in round-trip execution times, where it outperforms the monolithic setups in maximum result time, indicating its reliability in handling heavier workloads.

To summarize, the monolithic setups prioritize quick response times at lower request rates, whereas the Kubernetes cluster emphasizes scalability and consistency, especially under heavier loads and for round-trip executions. The choice between these configurations depends on factors such as expected workload, scalability requirements, and resource constraints, with the monolithic setups focusing on initial responsiveness and the Kubernetes cluster emphasizing scalability and reliability.

Table 1. Code Execution Engine Performance

	Monolith 4 Thread/8 Gb	Monolith 8 Thread/ 16Gb	Kubernetes Cluster 16 container/ 2 Nodes/ 16 GB
Min Response time	35.80ms	33.69ms	48.95ms
Max Response time	120.88ms	118.30ms	144.63ms
Average Response time @ 20 requests/sec	42.82ms	36.19ms	98.58ms
Average Response time @ D100 requests/sec	73.47ms	95.94ms	64.36ms
Min Result time (round-trip)	1sec	1sec	1sec
Max Result time (round-trip)	274sec	112sec	12sec
Average Result time @ 20 requests/sec	3sec	2.6sec	1sec
Average Result time @ 100 requests/sec	270.2sec	110.0sec	7sec

.

VIII. CONCLUSION

This paper presented the design and implementation details of a secure and scalable online coding assessment platform that leverages a microservice architecture and a robust technology stack. The platform empowers organizations with functionalities such as syndicate creation, secure assignment sharing, and efficient submission management, while prioritizing security through a multi-layered approach. The secure email sharing process with domain validation, microservice authentication, data encryption, and secure communication protocols ensure data protection and authorized access. The deployment strategy utilizing local Kubernetes for development and cloud-based deployments with autoscaling for production environments guarantees efficient development, testing, and optimal performance under varying loads.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 5, May 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1205332 |

REFERENCES

[1] A. Roy, G. Chowdhury, and S. R. Chowdhury, "A secure and scalable online coding assessment platform using microservices architecture," in 2023 18th International Conference on Cyber Warfare and Security (ICCWS), pp. 1-6, IEEE, 2023.

[2] B. Mistry, H. Parekh, K. Desai and N. Shah, "Online Examination System with Measures for Prevention of Cheating along with Rapid Assessment and Automatic Grading," 2022 5th International Conference on Advances in Science and Technology (ICAST), Mumbai, India, 2022, pp. 28-34.

[3] P. A. A. Dumal, W. K. D. Shanika, S. A. D. Pathinayake and T. C. Sandanayake, "Adaptive and automated online assessment evaluation system," 2017 11th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), Malabe, Sri Lanka, 2017, pp. 1-8.

[4] J. Huo, "A Lightweight Online Interactive Assessment Platform for SQL Teaching," 2022 IEEE 13th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2022, pp. 186-189.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details